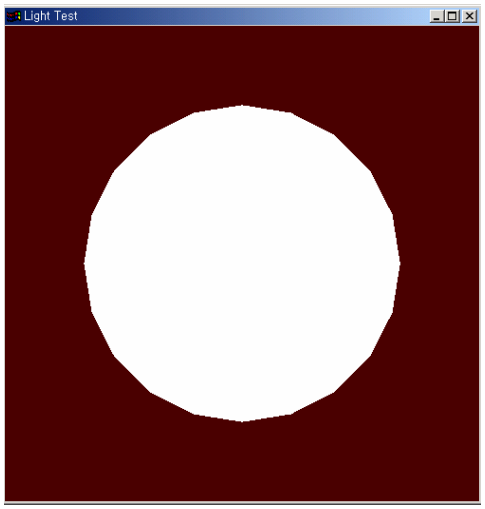
Graphics Programming

Lighting

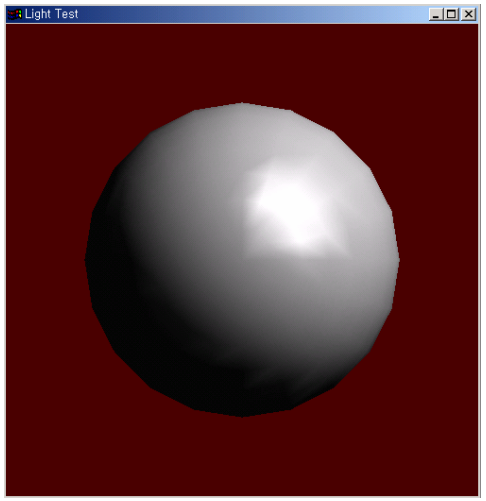
HyoungSeok Kim

- 3차원 그래픽스에 입체감 부여
  - Lighting (빛과 물체간의 상호 관계)에 의해서
  - 주어진 지점에서의 색상은 그 지점에서 반사되는 빛으로 대체

Viewing



Viewing + Lighting

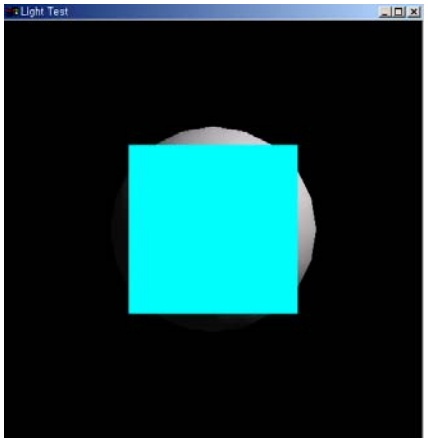


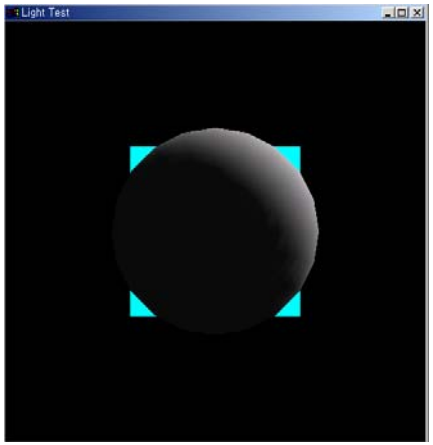
- 입체감 부여를 위한 구성 요소
  - Hidden-Surface Removal
    - Z-buffer Algorithm
  - Lighting
    - Light
      - Ambient, Diffuse, Specular
      - Position (Distant Light, One point Light)
    - Material
      - The reflection ratio of Ambient, Diffuse, Specular components
      - Emission

- 역할
  - 보이지 않는 면(hidden-surface)은 그리지 않음

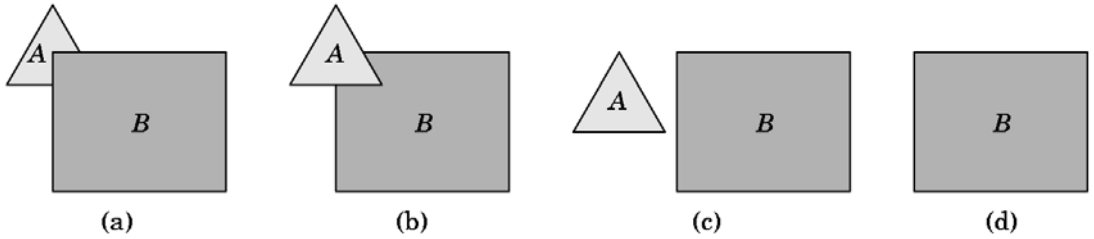
```
While(1) {
Get_viewing_point_from_mouse_position();
glClear(GL_COLOR_BUFFER_BIT);
Draw_3D_Object_A();
Draw_3D_Object_B();
}
```

```
glutInitDisplayMode(GLUT_DEPTH | … );
glEnable(GL_DPETH_TEST);
While(1) {
Get_viewing_point_from_mouse_position();
glClear(GL_COLOR_BUFFER_BIT);
Draw_3D_Object_A();
Draw_3D_Object_B();
}
```

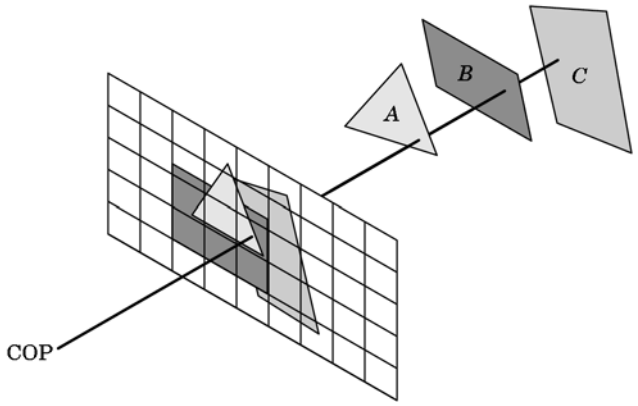




- object-space approaches
  - 3D 에서 face 간의 순서를 부여
  - painter's algorithm

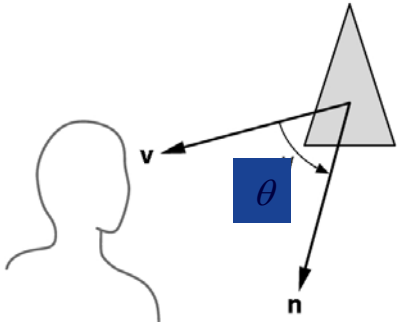


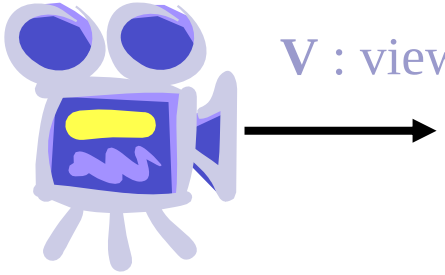
- image-space approaches
  - pixel 마다 보이는 물체를 찾음
  - z-buffer algorithm



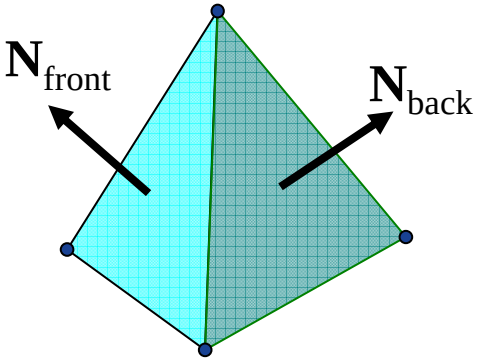
Back-Face Removal

- front face : camera 쪽으로 향한 face $-90° \le \theta \le 90°$
  - 화면에 나와야 한다 $N_{front} \cdot V < 0$
- back face : camera 반대 쪽을 향한 face $\theta < -90°, \theta > 90°$
  - 화면에 나오면 안 된다 $N_{back} \cdot V > 0$



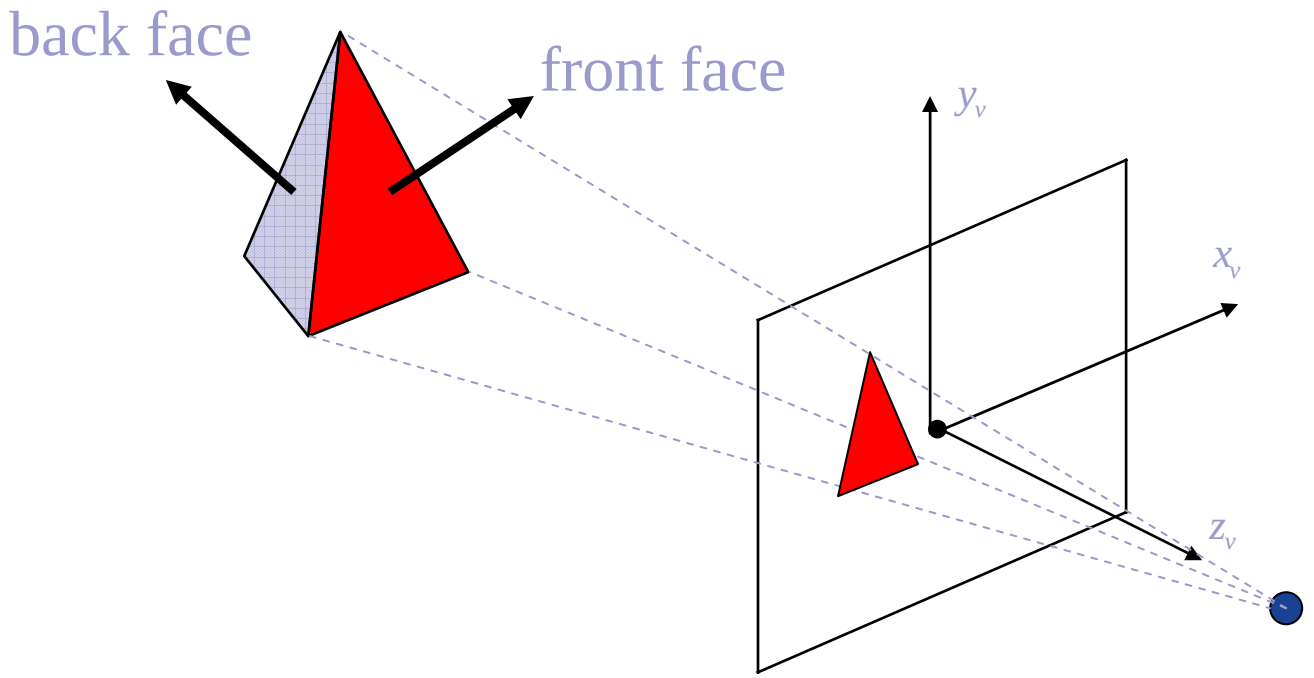


V : view vector

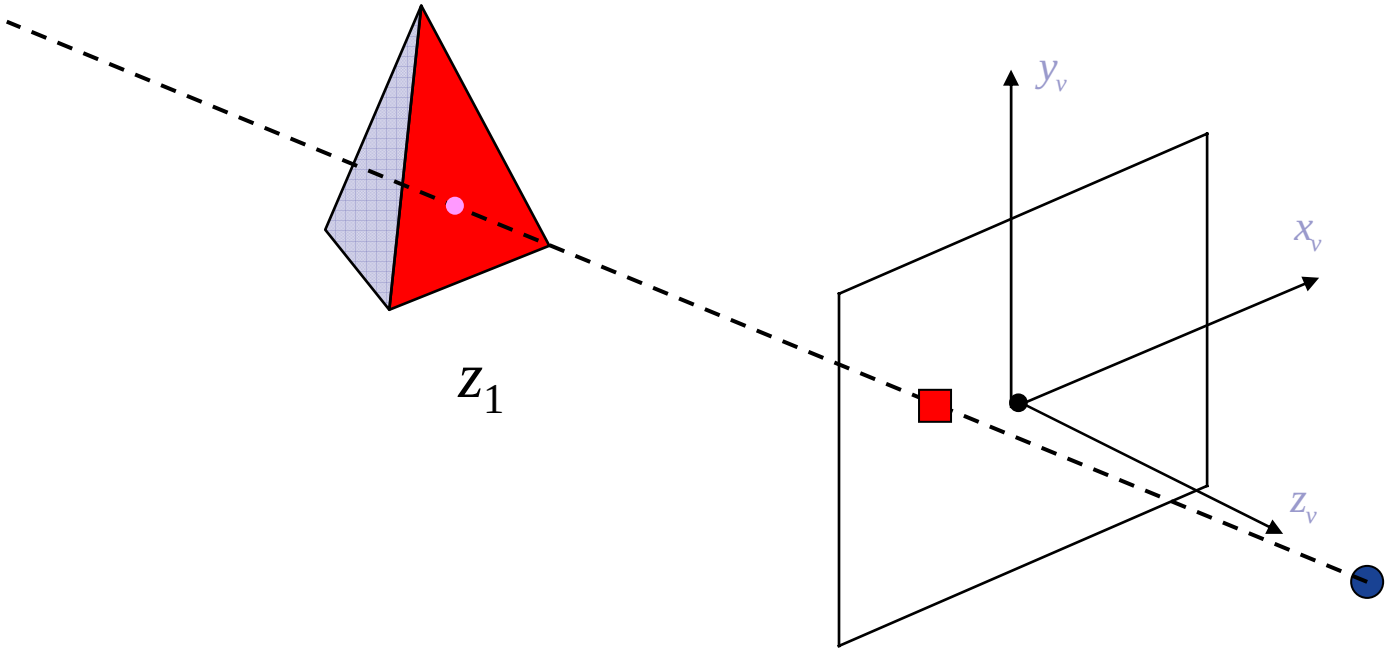


Back-Face Removal

- view coordinate system 에서는 back-face를 화면에 표시할 필요가 없다
  - 다른 면에 의해서 가려지므로

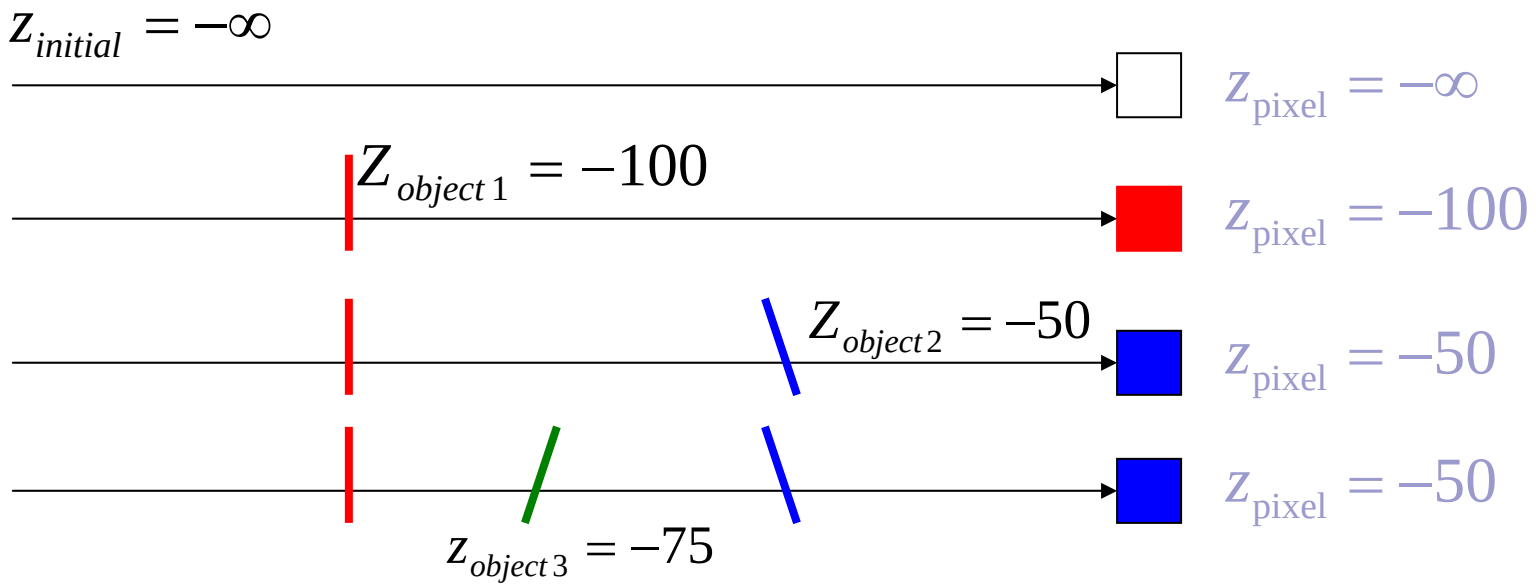


- 기본 아이디어
  - 화면 상의 pixel 하나를 그릴 때 마다 z 좌표도 함께 기억



Z-buffer Algorithm

- 초기 : 모든 pixel 의 z-좌표는 $-\infty$
- pixel 을 그릴 필요가 있으면,
  - if ($z_{pixel} < z_{object}$) then pixel update, $z_{pixel} = z_{object}$
  - else ignore



- coherence 의 이용
  - pixel 단위 처리일 때, 매번 다시 계산하지 않음
- two points on a polygon (x_1, y_1, z_1), (x_2, y_2, z_2)

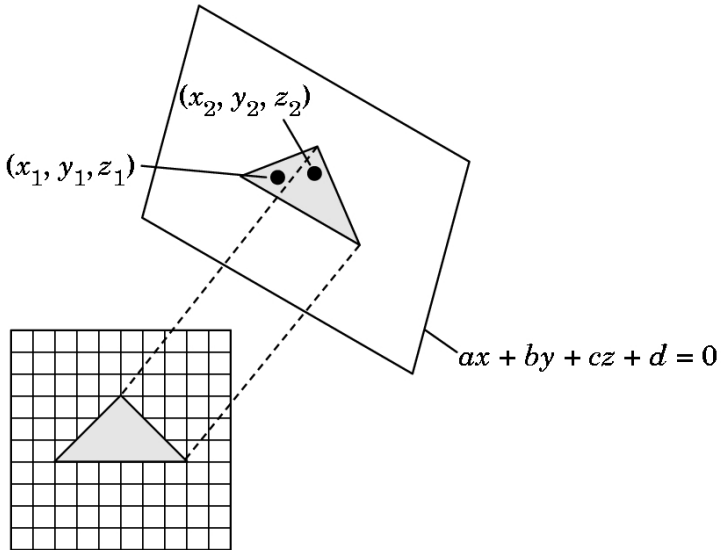
  $a\,x + b\,y + c\,z + d = 0$ plane equation

  $\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$, $\Delta z = z_2 - z_1$

  $a\,\Delta x + b\,\Delta y + c\,\Delta z = 0$
- 바로 옆 pixel 로 이동시,

  $\Delta x = 1$, $\Delta y = 0$

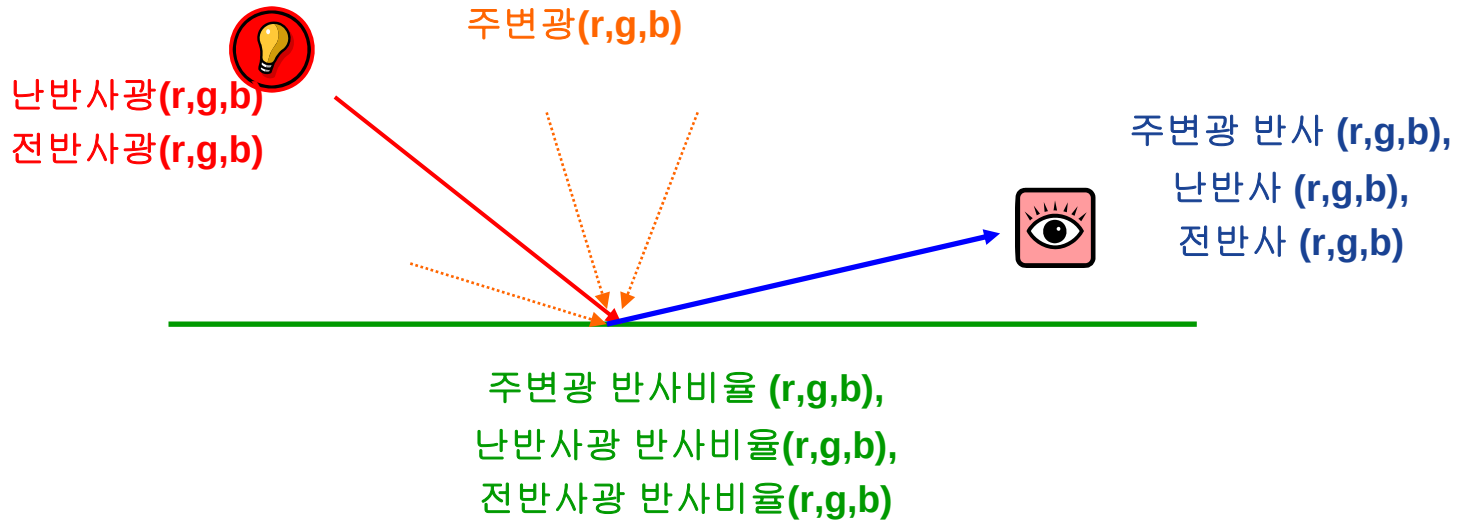
  $\Delta z = -(c/a)\,\Delta x$



Z-buffer Algorithm

- image space algorithm
  - 모든 물체는 일단 화면까지 projection 된다
- simple & efficient
  - 대부분의 video card 가 채택
  - OpenGL에서도 사용
- 단점 : 메모리가 많이 필요
  - pixel 하나 마다 (R, G, B, Z) 로 저장
    - Z 좌표를 저장하기 위한 메모리 필요

- 빛의 성분
  - 빨간색(R) + 초록색(G) + 파란색(B) $0.0 \le R, G, B \le 1.0$
- 빛의 종류
  - 주변광(Ambient) + 난반사광(Diffuse) + 전반사광(Specular)
- 광원의 색
  - 광원에서 방출하는 R, G, B 성분의 양에 따라 결정
- 표면의 재질
  - 표면에 들어왔다가 다양한 방향으로 반사되는 빛의 R, G, B 성분의 비율



Example


```
#include <gl/glut.h>   // (or others, depending on the system in use)

void init (void)
{
        GLfloat         light_position[] = { 1.0, 1.0, 1.0, 0.0};
        GLfloat         white_light[] = { 1.0, 1.0, 1.0, 1.0};
        GLfloat         model_ambient[] = { 0.1, 0.1, 0.1, 1.0};

        glClearColor (0.3, 0.0, 0.0, 0.0) ;
        glShadeModel(GL_SMOOTH);

        glLightfv(GL_LIGHT0, GL_POSITION, light_position);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, white_light);
        glLightfv(GL_LIGHT0, GL_SPECULAR,  white_light);
        glLightModelfv(GL_LIGHT_MODEL_AMBIENT, model_ambient);

        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);

        glEnable(GL_DEPTH_TEST);
}
```

Example


```
void display(void) {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
        glutSolidSphere(1.0, 20, 16);
        glutSwapBuffers();
        glFlush();
}

void reshape(int w, int h){
        glViewport(0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if ( w <= h )
                glOrtho(-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w, 1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
        else
                glOrtho(-1.5*(GLfloat)w/(GLfloat)h, 1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5,  -10.0, 10.0);
        glMatrixMode (GL_MODELVIEW);
        glLoadIdentity();
}
```

Example


```
void main (int argc, char** argv)
{
        glutInit (&argc, argv);
        glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowPosition (100, 100);
        glutInitWindowSize (500, 500);
        glutCreateWindow ("Light Test");
        init();

        glutDisplayFunc (display);
        glutReshapeFunc (reshape);
        glutMainLoop();
}
```

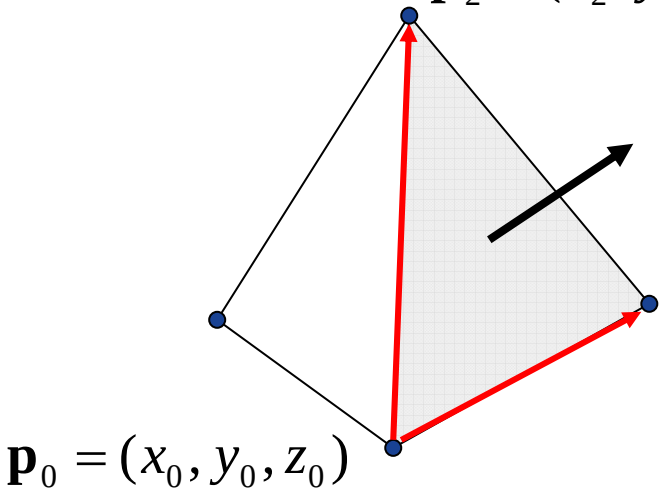
- 장면에 조명을 추가하는 단계
  1. 법선 벡터 계산 **(**물체의 각 정점마다**)**
     - 이웃한 면의 법선 벡터를 평균한다**.**
     - 면의 법선 벡터
       - 연결된 두 선분 벡터의 외적으로 계산
  2. 광원 속성 지정 및 설치
     - 광원의 **Ambient, Diffuse, Specular** 성분의 **r, g, b** 값 부여
     - 광원의 위치 지정
     - 광원의 종류에 따른 추가 속성 지정 **(Spot Light)**
  3. 광원 사용 여부 선언
     - **glEnable(GL_LIGHTING) :** 광원 활성화 → 전원 공급
     - **glEnable(GL_LIGHT0) : 0**번 광원 활성화 → 스위치 **On**
     - **8** 개의 광원 사용 가능

Normal Vector Computation

- 면의 법선 벡터
  - 면의 방정식이 주어진 경우

$$ax + by + cz + d = 0 \qquad \mathbf{n} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

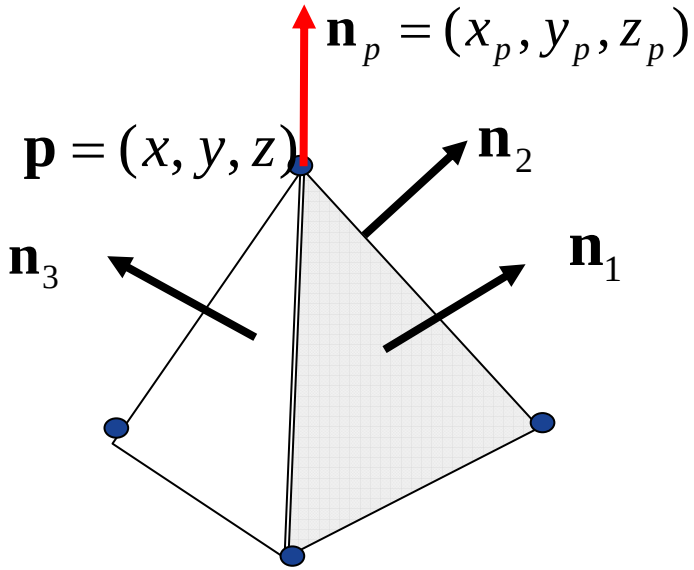
  - 정점의 좌표가 주어진 경우



$$\mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0) = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 - x_0 & y_1 - y_0 & z_1 - z_0 \\ x_2 - x_0 & y_2 - y_0 & z_2 - z_0 \end{vmatrix}$$

Normal Vector Computation

- 정점의 법선 벡터




```
glBegin(GL_POLYGON);
    glNormal3f (x_p, y_p, z_p);
    glVertex3f(x, y, z);
...
glEnd();
```

$$\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a & b & c \\ d & e & f \end{vmatrix} = (bf - ce)\mathbf{i} + (-1)(af - cd)\mathbf{j} + (ae - bd)\mathbf{k}$$

광원 속성 함수 glLight

- glLightf (light, pname, param);
- glLightfv (light, pname, *param);
  - light = GL_LIGHT0, GL_LIGHT1, … GL_LIGHT7
  - pname = 속성
  - param = 속성값, *param = 속성값 벡터
- Pname
  - GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR
  - GL_POSITION
    - Light_position1[] = { 1.0, 2.0, 3.0, 1.0}; 점 광원 = 위치
    - Light_position2[] = { 1.0, 2.0, 3.0, 0.0}; 원거리 광원 = 방향
    - glLightfv(GL_LIGHT0, GL_POSITION, Light_position1);
  - GL_SPOT_DIRECTION, GL_SPOT_EXPONENT
  - GL_SPOT_CUTOFF
    - glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 30.0);
  - GL_CONSTANT_ATTENUATION
  - GL_LINEAR_ATTENUATION
  - GL_QUADRATIC_ATTENUATION

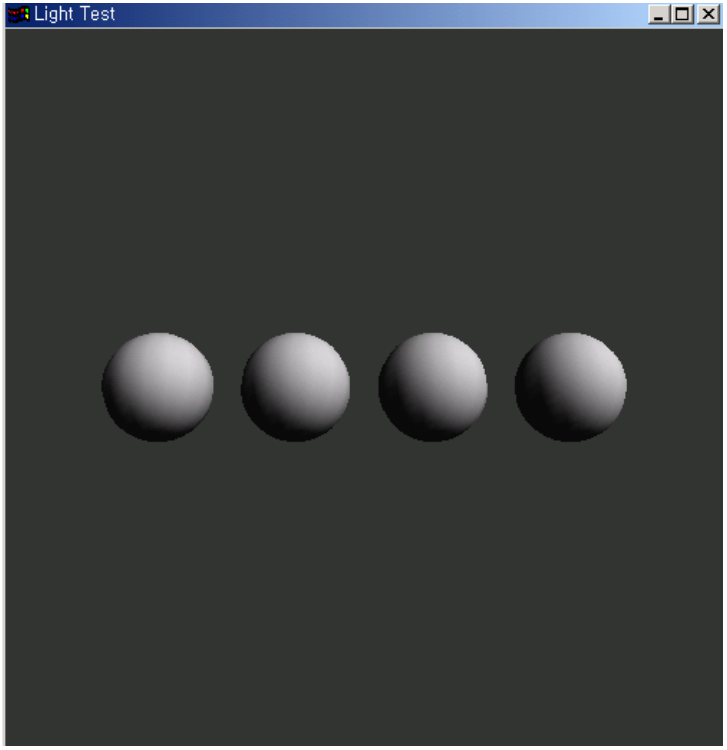
전역 변수 설정

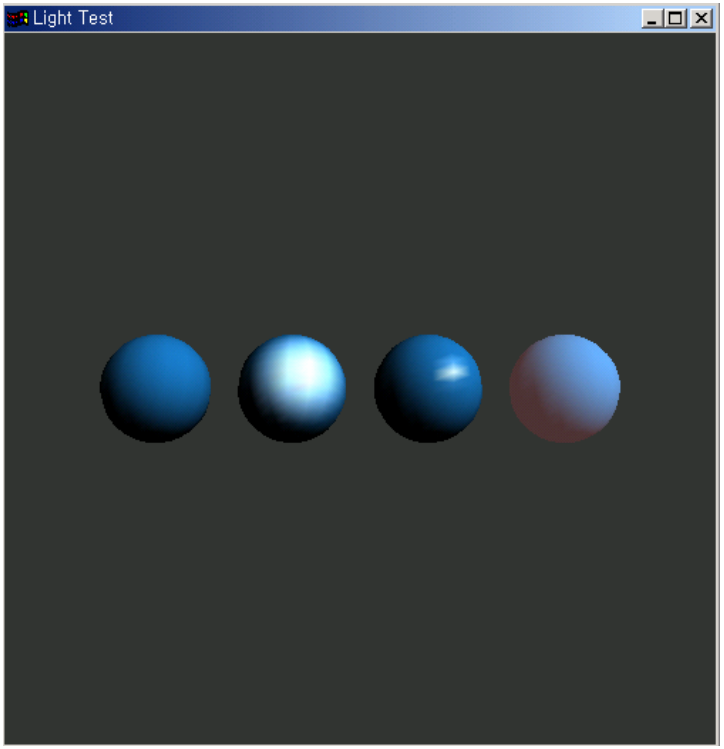
- 라이팅 모델 선택
  - 전역적 **ambient** 라이트 세기
    - **glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambient);**
  - 시점의 위치
    - **glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_FALSE);**
  - 라이팅 적용 면 선택 : 양면 **or** 단면
    - **glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_FALSE);**
  - **Specular** 색상을 **Texture mapping** 후에 적용 여부
    - **glLightModeli(GL_LIGHT_MODEL_COLOR_CONTROL, GL_SINGLE_COLOR);**
      - **AMBIENT, DIFFUSE** 와 함께 계산
    - **glLightModeli(GL_LIGHT_MODEL_COLOR_CONTROL, GL_SEPARATE_SPECULAR_COLOR);**
      - **SPECULAR** 하이라이트 효과를 텍스쳐 매핑 된 후에 적용

재질 속성 함수 glMaterial

- glMaterialf (face, pname, param);
- glMaterialfv (face, pname, *param);
  - face = GL_FRONT, GL_ BACK, GL_FRONT_AND_BACK
- Pname
  - GL_AMBIENT, GL_DIFFUSE, GL_AMBIENT_AND_DIFFUSE,
  - GL_SPECULAR
    - glMaterialfv(GL_BACK, GL_SPECULAR, mat_specualr);
  - GL_SHININESS
    - glMaterialf(GL_FRONT, GL_SHININESS, 100);
  - GL_EMISSION, GL_COLOR_INDEXES
- Color 색상 변경
  - glColorMaterial(face, mode);
    - face : GL_FRONT, GL_ BACK, GL_FRONT_AND_BACK
    - mode : GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_AMBIENT_AND_DIFFUSE
    - 예제) glEnable(GL_COLOR_MATERIAL);
    - glColorMaterial(GL_FRONT, GL_DIFFUSE);
    - glColor3f(1.0, 1.0, 0.0);
    - glDisable(GL_COLOR_MATERIAL);

Example





Example


```
#include <gl/glut.h>

GLfloat     light_ambient0[] = { 0.0, 0.0, 0.0, 1.0};
GLfloat     light_diffuse0[] = { 1.0, 1.0, 1.0, 1.0};
GLfloat     light_specular0[] = { 1.0, 1.0, 1.0, 1.0};
GLfloat     light_position0[] = { 1.0, 1.0, 1.0, 0.0};

GLfloat     light_ambient1[] = { 1.0, 0.0, 0.0, 1.0};
GLfloat     light_diffuse1[] = { 1.0, 0.0, 0.0, 1.0};
GLfloat     light_specular1[] = { 1.0, 0.0, 0.0, 1.0};
GLfloat     light_position1[] = { 2.0, 2.0, 2.0, 1.0};
GLfloat     spot_direction[] = {-1.0, -1.0, 0.0};

GLfloat     no_mat[] = {0.0, 0.0, 0.0, 1.0};
GLfloat     mat_ambient[] = { 0.7, 0.7, 0.7, 1.0};
GLfloat     mat_ambient_color[] = { 0.8, 0.8, 0.2, 1.0};
GLfloat     mat_diffuse[] = { 0.1, 0.5, 0.8, 1.0};
GLfloat     mat_specular[] = { 1.0, 1.0, 1.0, 1.0};
GLfloat     no_shininess[] = { 0.0};
GLfloat     low_shininess[] = {5.0};
GLfloat     high_shininess[] = {100.0};
GLfloat     mat_emission[] = { 0.3, 0.2, 0.2, 0.0};
```

Example


```
void init (void) {
        glClearColor (0.0, 0.0, 0.0, 0.0) ;
        glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient0);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse0);
        glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular0);
        glLightfv(GL_LIGHT0, GL_POSITION, light_position0);

        glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient1);
        glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse1);
        glLightfv(GL_LIGHT1, GL_SPECULAR, light_specular1);
        glLightfv(GL_LIGHT1, GL_POSITION, light_position1);
        glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, 0.5);
        glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.1);
        glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, 0.1);

        glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 45.0);
        glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction);
        glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.0);
        glShadeModel(GL_SMOOTH);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glEnable(GL_DEPTH_TEST);
}
```

Example


```
void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

	glPushMatrix();
		glTranslatef(-3.75, 0.0, 0.0);
		glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
		glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
		glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
		glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
		glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
		glutSolidSphere(1.0, 16, 16);
	glPopMatrix();

	glPushMatrix();
		glTranslatef(-1.25, 0.0, 0.0);
		glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
		glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
		glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
		glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
		glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
		glutSolidSphere(1.0, 16, 16);
	glPopMatrix();
```

Example


```
glPushMatrix();
        glTranslatef(1.25, 0.0, 0.0);
        glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
        glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
        glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
        glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
        glutSolidSphere(1.0, 16, 16);
    glPopMatrix();

    glPushMatrix();
        glTranslatef(3.75, 0.0, 0.0);
        glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
        glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
        glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
        glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
        glutSolidSphere(1.0, 16, 16);
    glPopMatrix();

    glutSwapBuffers();
    glFlush();
}
```

Example


```
void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(45.0, 1.0, 1.0, 200.0);
        glMatrixMode (GL_MODELVIEW);
        glLoadIdentity();
        gluLookAt(0.0, 15.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}

void main (int argc, char** argv)
{
        glutInit (&argc, argv);
        glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowPosition (100, 100);
        glutInitWindowSize (500, 500);
        glutCreateWindow ("Light Test");
        init();
        glutDisplayFunc (display);
        glutReshapeFunc (reshape);
        glutMainLoop();
}
```